

EIE

Escuela de
Ingeniería Eléctrica

Universidad de Costa Rica

Facultad de Ingeniería
Escuela de Ingeniería Eléctrica



UNIVERSIDAD DE
COSTA RICA

IE-0117

Programación Bajo Plataformas Abiertas

Laboratorio #3

Fernando Jiménez Ureña B74020

II ciclo
Septiembre 2021

1. Resumen

En el presente laboratorio, se desarrollarán tres diferentes temas esenciales para poder desenvolverse de mejor manera en el sistema operativo Linux. Primero, se desarrollará el tema que corresponde a la gestión de Usuarios, Grupos y Permisos en Linux. Luego, se realizará una investigación sobre las funciones de administradores de tareas como lo son 'Cron', 'Crontab' así como los comandos de 'rsync' y el protocolo 'ssh'. Finalmente, se procederá a llevar a cabo la descarga de un programa desde su código fuente así como la explicación de los respectivos pasos para lograrlo.

2. Nota Teórica

En esta sección se explicarán conceptos esenciales para poder entender correctamente el desarrollo del laboratorio. Para poder realizar muchos de los pasos que se llevarán a cabo durante el laboratorio, estos deben realizarse desde el usuario 'root' o mediante el uso de permisos 'sudo'. En Linux, existe un usuario administrador o superusuario, que es el que posee el nivel más elevado de privilegios (usuario 'root'), y el resto de usuarios, con un nivel de permisos mucho más restringido. Para poder usarlo, se escribe en la terminal del sistema 'su -' y se procede a ingresar la contraseña del administrador. También existen otros mecanismos para poder otorgarle permisos de administrador a cualquier usuario, para eso existe el comando 'sudo'. Dicho comando se debe instalar antes en el sistema, y también ingresar dentro de la configuración del comando cuales usuarios tendrán permisos para poder utilizarlo. [1]

Para poder crear y gestionar usuario, grupos y permisos, los cambios deben realizarse usando el superusuario o haciendo uso del comando 'sudo'. Linux es un sistema multiusuario, concebido teniendo en mente que el sistema sería utilizado por múltiples usuarios. Linux posee una importante arquitectura de permisos, la cuál es uno de los pilares sobre el que descansa la seguridad del sistema.

En Linux, todos los archivos y directorios tienen asociado un conjunto de permisos que van asociados a grupos y usuarios. Estos permisos pueden ser de lectura, escritura o ejecución. Existen tres tipos de usuarios en Linux: Superusuario que ya fue explicado anteriormente y que posee todos los privilegios sobre el sistema, luego existen los usuarios del sistema que son aquellos que están específicamente vinculados a ciertos servicios como por ejemplo 'bin', 'apache', 'clamav', 'pulse' y entre otros; estos usuarios generalmente se crean automáticamente en la instalación del sistema. Finalmente existen los usuarios estándar que son aquellos donde se almacenan los archivos personales, preferencias variadas aplicaciones, archivos temporales y entre otros. [2]

Existe otro concepto importante que es el de Grupos. Una de las maneras para simplificar la asignación de permisos entre tantos usuarios es por medio del método de Grupos. Por lo tanto, a la hora de especificar ciertos permisos sobre un conjunto de cuentas se puede establecer dichos permisos directamente sobre el Grupo. Cada grupo se identifica por un Group ID. Todos los ficheros en Linux son propiedad de un usuario y de un grupo; y cada uno de ellos posee privilegios y permisos específicos. [2]

En los archivos y directorios se pueden asignar tres tipos diferentes de permisos: Lectura, Escritura y Ejecución. Los permisos de lectura se representan con la letra 'r' y estos dan la posibilidad de acceder a un archivo o directorio y poder leer su contenido. Por otro lado, los permisos de Escritura se representan con la letra 'w' y dan la capacidad de borrar o añadir archivos dentro del directorio o archivo. Finalmente los permisos más importantes son los de ejecución y se representan con la letra 'x' ya que permiten ejecutar un determinado archivo en el sistema.

Para esta sección del laboratorio, se utilizaran distintos comandos y entre ellos destacan:

- **adduser:** Este comando tiene como función recibir un parámetro el nombre de un nuevo usuario o agregar un usuario a un grupo en específico.
- **chown:** Este comando se utiliza para poder cambiar el propietario del archivo e incluso el grupo propietario
- **su:** La principal función del comando es cambiar el usuario dentro de la terminal. Si se le agrega el caracter - seguido del comando su, se accede al usuario 'root' del sistema.
- **exit:** Este comando se utiliza para salir del usuario 'root'
- **chmod:** Este comando es de gran utilidad ya que es el encargado de cambiar los permisos de lectura, escritura y ejecución de los archivos y directorios además del grupo al cual pertenece.
- **addgroup** Este comando se utiliza para crear un nuevo grupo
- **add (usuario) (grupo):** Este comando se utiliza para agregar a algún usuario a un grupo
- **groups:** Este comando imprime los grupos al que el usuario utilizando la terminal en ese preciso momento pertenece
- **chgrp:** Este comando se utiliza para cambiar al grupo dueño de un directorio o archivo
- **deluser:** Este comando se utiliza para eliminar el nombre de un usuario o un grupo
- **apt moo:** Este comando imprime un dibujo de una vaca en la terminal

Estos son algunos de los comandos más relevantes que se utilizarán en el transcurso del laboratorio, sin embargo hay más pero ya fueron descritos con anterioridad en pasados laboratorios.

El Cron, es un administrador de procesos en segundo plano que ejecuta procesos a intervalos regulares es decir cada minuto, cada día, semana o mes. Por otro lado, otra de las definiciones importantes que se deben de conocer es la del Crontab. El Crontab es un archivo de texto que guarda la lista de comandos que se desean ejecutar en el tiempo especificado por el usuario. Básicamente es una forma de administrar las tareas del Cron. [3]

Otro comando importante a tomar en cuenta es el Rsync. En Linux es una herramienta muy útil ya que transfiere y sincroniza archivos o directorios de manera eficiente entre una máquina local, un servidor remoto o cualquiera de estos, es un importante comando para mejorar la productividad. [4]

El comando `rsync` en conjunto a `ssh` es una combinación muy poderosa ya que permite que se puedan compartir archivos o directorios mediante los cuales se sincronizan de servidores remotos o locales. [5]

Para que diferentes archivos o directorios se sincronicen puede conllevar mucho tiempo por lo que la utilidad del comando `rsync` puede agregar excelentes funciones para ahorrar tiempo, hasta si se pierde la conexión a red, esta herramienta continua exactamente donde se queda una vez que la conexión se restablezca. Un punto importante para utilizar este comando es asegurarse que se encuentre instalado en el sistema, de no ser así se procede a ingresar `sudo apt-get install rsync`. En relación al `ssh` se pueden compartir archivos o directorios mediante los cuales se sincronizan de servidores remotos o locales. [5]

Finalmente, en la última parte del laboratorio se lleva a cabo el proceso de instalación desde código fuente de NodeJs. Es importante conocer qué es el código fuente. El código fuente es el código madre de cualquier archivo donde se puede usar o modificar su código para mejorarlo o adaptarlo a las necesidades del usuario en cualquier momento. Por otro lado, el programa NodeJS es un tiempo de ejecución de JavaScript que se utiliza para programar desde el lado del servidor. [6]

3. Análisis de Resultados

3.1. Usuarios, grupos y permisos

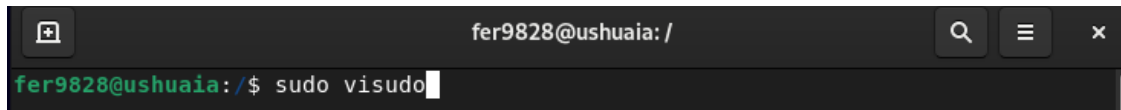
En la primera parte del Laboratorio, se debe crear un nuevo usuario llamado 'Labo3'. En la Figura 1., se puede apreciar que utilizando el comando `sudo` en combinación con el comando 'adduser' se crea un nuevo usuario llamado 'labo3'. A este usuario también se le configura una contraseña.

```
fer9828@ushuaia:~$ sudo adduser labo3
[sudo] password for fer9828:
Adding user `labo3' ...
Adding new group `labo3' (1001) ...
Adding new user `labo3' (1001) with group `labo3' ...
Creating home directory `/home/labo3' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for labo3
Enter the new value, or press ENTER for the default
  Full Name []: Labo3
   Room Number []:
   Work Phone []:
   Home Phone []:
    Other []:
Is the information correct? [Y/n] y
fer9828@ushuaia:~$
```

Figura 1: Creación del usuario 'Labo3'

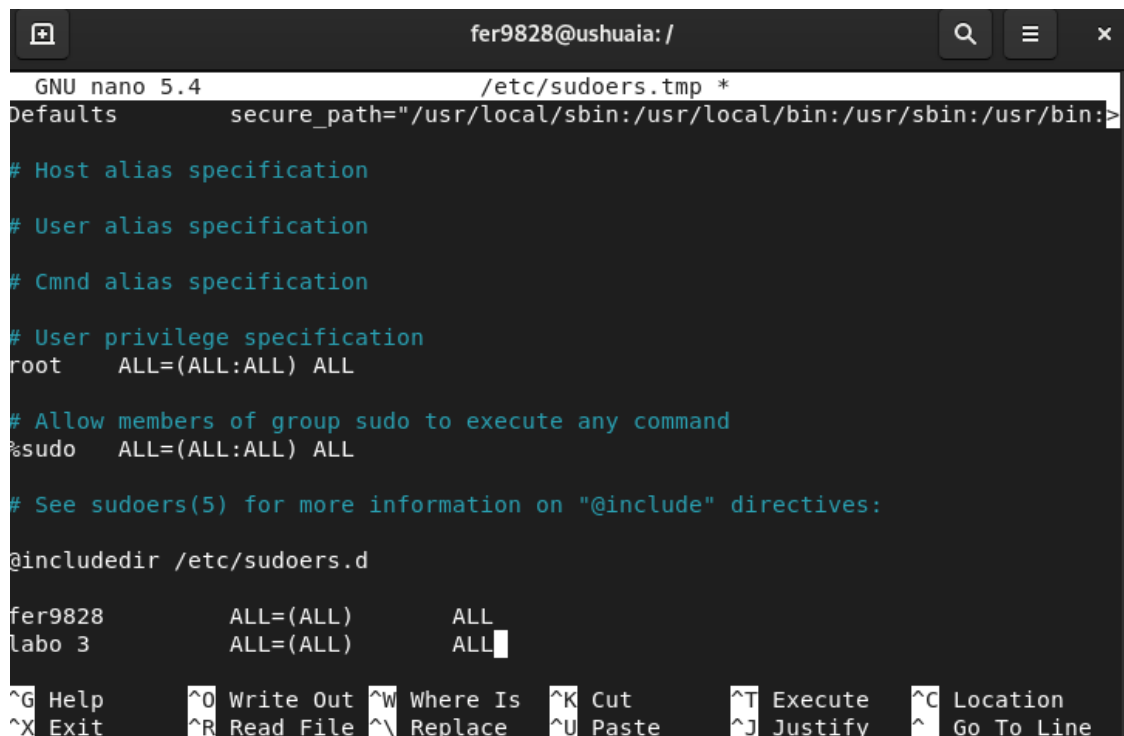
Antes de poder continuar, se le deben agregar los permisos elevados de 'sudo' al usuario recién creado. Esto con el objetivo de poder realizar cambios necesarios solicitados en el laboratorio como es el caso de crear nuevos directorios y entre otros.

Para poder agregar estos permisos se utiliza el comando 'visudo' desde el usuario 'root' o desde algún usuario con permisos sudo como lo fue en este caso que se accedió desde el usuario 'fer9828'. Desde el comando 'visudo' se puede editar el fichero /etc/sudoers y se agrega el usuario 'labo3' en la sección de usuarios con permisos sudo. Los pasos realizados se pueden apreciar en la Figura 2. y la Figura 3.



```
fer9828@ushuaia:/$ sudo visudo
```

Figura 2: Permisos sudo para el usuario 'Labo3'



```
GNU nano 5.4 /etc/sudoers.tmp *
Defaults    secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:>
# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root    ALL=(ALL:ALL) ALL
# Allow members of group sudo to execute any command
%sudo    ALL=(ALL:ALL) ALL
# See sudoers(5) for more information on "@include" directives:
@includedir /etc/sudoers.d

fer9828    ALL=(ALL)    ALL
labo 3    ALL=(ALL)    ALL
```

Figura 3: Permisos sudo para el usuario 'Labo3'

Ahora se procede a cambiar de usuario desde 'fer9828' a 'labo3' mediante el comando 'su'. Además ya una vez dentro del usuario 'labo3', se procede a moverse entre directorios hasta llegar al directorio \$HOME mediante el comando 'cd'. Ya una vez dentro del directorio \$HOME, se procede a crear un directorio haciendo uso del comando 'mkdir' y seguido el nombre del directorio que se desea crear, en este caso 'PruebasPermisos'. Como se puede apreciar en la Figura 4. este proceso fue completado con éxito. Haciendo uso del comando

'ls -l' y el comando 'pwd' se puede comprobar que el directorio fue creado con éxito y en el directorio del usuario creado.

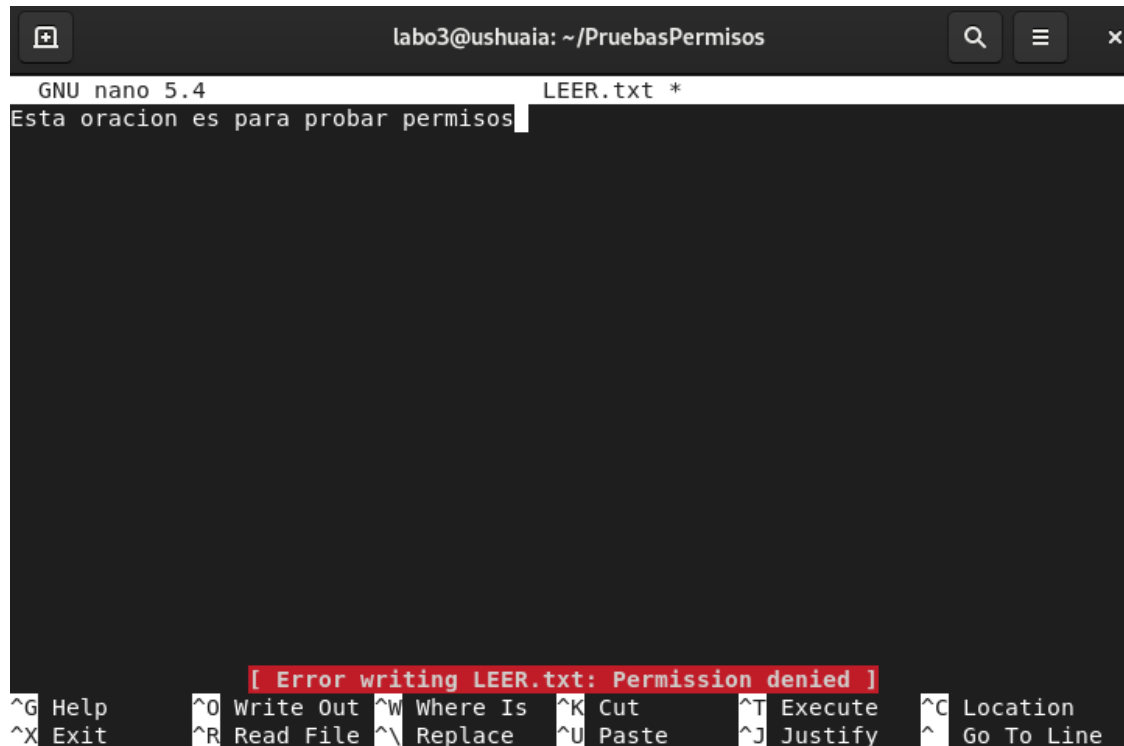
```
fer9828@ushuaia:/$ su labo3
Password:
labo3@ushuaia:/$ cd home/labo3
labo3@ushuaia:~$ pwd
/home/labo3
labo3@ushuaia:~$ sudo mkdir PruebasPermisos
[sudo] password for labo3:
labo3@ushuaia:~$ ls
PruebasPermisos
labo3@ushuaia:~$ ls -l
total 4
drwxr-xr-x 2 root root 4096 Sep 23 14:55 PruebasPermisos
labo3@ushuaia:~$
```

Figura 4: Creación del directorio 'PruebasPermisos'

Se puede observar en la Figura 4. que en el recién creado directorio, el usuario 'labo3' no tiene permisos para realizar ningún cambio (lectura, escritura y ejecución). Esto quiere decir que cuando se intente crear el archivo LEER.txt desde el usuario 'labo3' la acción será denegada. Se puede apreciar en la Figura 5. y en la Figura 6. que se realizó dicha prueba desde el editor de texto 'nano' y que efectivamente los permisos para esta acción fueron denegados.

```
labo3@ushuaia: ~/PruebasPermisos
labo3@ushuaia:~$ cd PruebasPermisos/
labo3@ushuaia:~/PruebasPermisos$ nano LEER.txt
```

Figura 5: Creación del archivo .txt

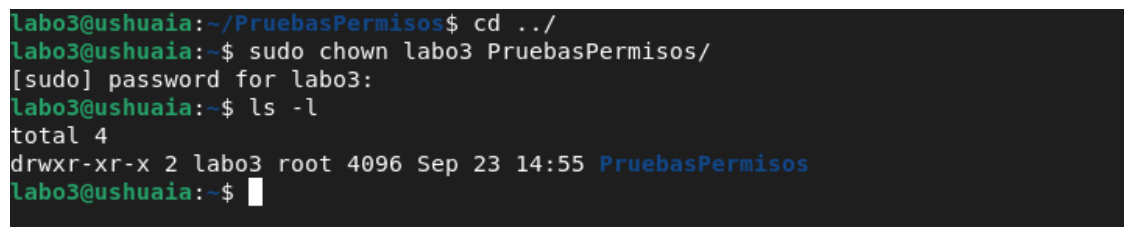


```
labo3@ushuaia: ~/PruebasPermisos
GNU nano 5.4 LEER.txt *
Esta oracion es para probar permisos

[ Error writing LEER.txt: Permission denied ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Figura 6: Creación del archivo .txt

Para poder realizar cambios en el directorio 'PruebasPermisos' desde el usuario 'labo3', se deben cambiar los permisos en dicho directorio. Para realizar esto, se utiliza el comando 'chown' el cuál le cambia la pertenencia del directorio al usuario 'labo3'. Dicho procedimiento se puede apreciar en la Figura 7.



```
labo3@ushuaia:~/PruebasPermisos$ cd ../
labo3@ushuaia:~$ sudo chown labo3 PruebasPermisos/
[sudo] password for labo3:
labo3@ushuaia:~$ ls -l
total 4
drwxr-xr-x 2 labo3 root 4096 Sep 23 14:55 PruebasPermisos
labo3@ushuaia:~$
```

Figura 7: Permisos otorgados del directorio al usuario 'labo3'

Luego de este cambio, se puede proceder a crear el archivo LEER.txt dentro del directorio 'PruebasPermisos'. Para realizar este proceso, se abre el editor de texto 'nano' y se escriben las líneas de texto que se desean guardar. En la Figura 8. se puede apreciar la línea de texto guardada en el archivo LEER.txt

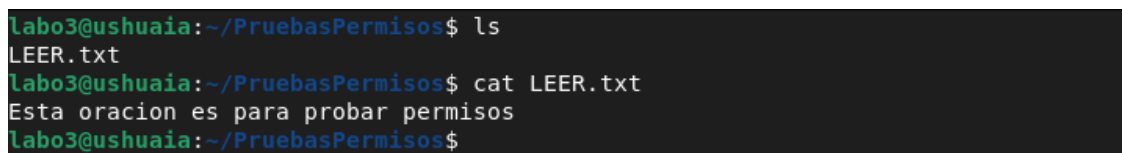


```
GNU nano 5.4 LEER.txt *
Esta oracion es para probar permisos

File Name to Write: LEER.txt
^G Help      M-D DOS Format  M-A Append     M-B Backup File
^C Cancel    M-M Mac Format  M-P Prepend    ^T Browse
```

Figura 8: Línea de texto guardada en el archivo .txt

Seguidamente, desde la terminal se puede comprobar que el archivo LEER.txt fue creado correctamente dentro del directorio 'PruebasPermisos'. Y luego, como se aprecia en la Figura 9. mediante el comando 'cat' se imprime el texto contenido dentro el archivo LEER.txt. Se puede apreciar que el procedimiento realizado fue un éxito.



```
labo3@ushuaia:~/PruebasPermisos$ ls
LEER.txt
labo3@ushuaia:~/PruebasPermisos$ cat LEER.txt
Esta oracion es para probar permisos
labo3@ushuaia:~/PruebasPermisos$
```

Figura 9: Línea de texto contenida en el archivo LEER.txt

Luego, siguiendo las instrucciones del Laboratorio, se deben cambiar los permisos del directorio 'PruebasPermisos' con el uso de un solo comando. Para completar este paso, se hizo uso del comando 'chmod'. Usando este comando, se le otorgaron los permisos faltantes (permisos de escritura) al usuario 'labo3' y al grupo al cual pertenece mediante el agregado 'g+w' y luego también se hace el agregado 'o-w' que significa que cualquier usuario que no esté en el grupo no podrá tener acceso a ninguno de los permisos de lectura, escritura ni ejecución dentro del directorio 'PruebasPermisos'


```
labo3@ushuaia:~/PruebasPermisos$ cd ../
labo3@ushuaia:~$ ls -l
total 4
drwxr-xr-x 2 labo3 root 4096 Sep 23 17:02 PruebasPermisos
labo3@ushuaia:~$ chmod g+w,o-rx PruebasPermisos/
labo3@ushuaia:~$ ls -l
total 4
drwxrwx--- 2 labo3 root 4096 Sep 23 17:02 PruebasPermisos
labo3@ushuaia:~$
```

Figura 10: Permisos en el directorio 'PruebasPermisos'

Luego, se realiza la prueba volviendo al usuario 'fer9828' e intentando entrar al directorio 'PruebasPermisos'. Se puede observar en la Figura 11., que si se intenta entrar al directorio se muestra un mensaje de Permisos Denegados. Luego en la Figura 12. se intenta entrar directo al archivo LEER.txt y se puede demostrar que se muestra el mismo mensaje de Permisos Denegados y además es interesante notar que tampoco se pueden leer las líneas que se habían escrito antes. Esto se debe a que como fue mencionado antes, se quitaron todos los permisos de escritura, lectura y ejecución al directorio 'PruebasPermisos' a cualquier usuario que no sea al usuario 'labo3' o alguien de su grupo.

```
labo3@ushuaia:~$ su fer9828
Password:
fer9828@ushuaia:/home/labo3$ ls
PruebasPermisos
fer9828@ushuaia:/home/labo3$ cd PruebasPermisos/
bash: cd: PruebasPermisos/: Permission denied
fer9828@ushuaia:/home/labo3$ nano PruebasPermisos/LEER.txt
```

Figura 11: Pruebas en el directorio 'PruebasPermisos'

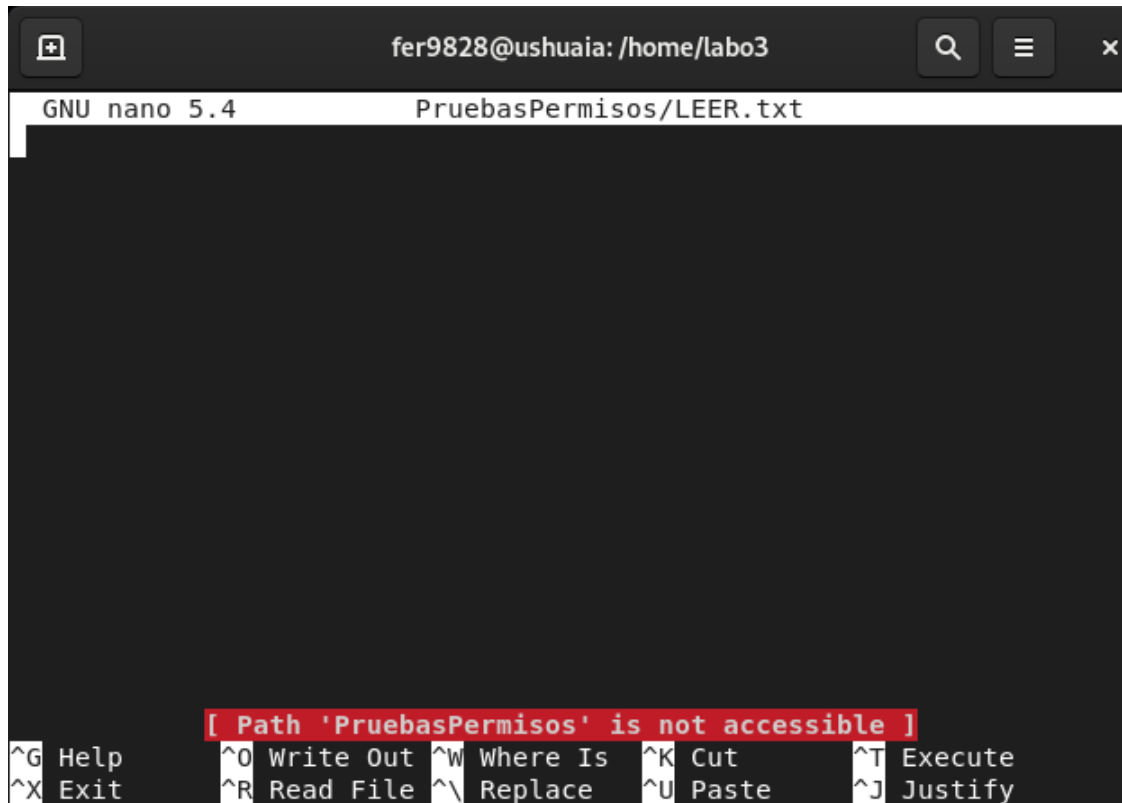


Figura 12: Pruebas en el directorio 'PruebasPermisos'

Seguidamente se procede a crear un nuevo grupo llamado 'grupolabo3'. Para esto primero se accede al usuario root del sistema y desde ahí se utiliza el comando 'groupadd'. Este comando se utiliza para crear un grupo nuevo en el sistema y el argumento 'grupolabo3' significa el nombre del grupo que se está creando. Ya con el grupo creado, se procede a añadir los usuarios 'labo3' y 'fer9828' a dicho grupo utilizando el comando 'usermod' cuya función es realizar varios cambios en la configuración de los usuarios del sistema. En este caso se utilizan dos argumentos luego del comando: '-a' que quiere decir que es añadir a un grupo y luego '-G' que significa que al grupo al cual se está añadiendo será un grupo secundario en el usuario. Este procedimiento realizado se puede apreciar en la Figura 13.

Cabe destacar que cuando un usuario es creado, este es automáticamente asignado a un grupo primario con el mismo nombre que el del usuario. Por lo tanto este argumento '-G' lo que hace es añadir al usuario a un grupo secundario que se debe especificar, en este caso 'grupolabo3' [7]

```
fer9828@ushuaia:/home/labo3$ su -  
Password:  
root@ushuaia:~# groupadd grupolabo3  
root@ushuaia:~# usermod -a -G grupolabo3 labo3  
root@ushuaia:~# usermod -a -G grupolabo3 fer9828  
root@ushuaia:~#
```

Figura 13: Creación del grupo 'grupolabo3'

Para revisar a cuales grupos pertenecen los usuarios 'fer9828' y 'labo3', se puede utilizar el comando 'groups' desde el usuario root. En la Figura 14. se puede notar que tanto el 'fer9828' como 'labo3' se encuentran en el 'grupolabo3'. Además se puede apreciar que el usuario 'fer9828' pertenece a otros grupos.

```
root@ushuaia:~# groups fer9828  
fer9828 : fer9828 cdrom floppy audio dip video plugdev netdev bluetooth scanner grupolabo3  
root@ushuaia:~# groups labo3  
labo3 : labo3 grupolabo3  
root@ushuaia:~#
```

Figura 14: Grupos a los que pertenecen los usuarios 'fer9828' y 'labo3'

Luego, se debe ejecutar el comando `exec su -l $USER` en la terminal, donde en lugar de escribir '\$USER', se escribió el nombre del usuario 'fer9828'. Se ejecutó dicho comando como se observa en la Figura 15. y se puede apreciar que no ocurrió ningún cambio y que efectivamente ambos usuarios: *fer9828* y *labo3* seguían perteneciendo a los mismos grupos.

El comando `exec` se utiliza para ejecutar un comando del sistema sin embargo una vez ejecutado ya no se puede retornar al programa inicial. [8]

```
root@ushuaia:~# groups fer9828  
fer9828 : fer9828 cdrom floppy audio dip video plugdev netdev bluetooth scanner  
grupolabo3  
root@ushuaia:~# groups labo3  
labo3 : labo3 grupolabo3  
root@ushuaia:~# exec su -l fer9828  
fer9828@ushuaia:~$ groups fer9828  
fer9828 : fer9828 cdrom floppy audio dip video plugdev netdev bluetooth scanner  
grupolabo3  
fer9828@ushuaia:~$ groups labo3  
labo3 : labo3 grupolabo3  
fer9828@ushuaia:~$
```

Figura 15: Ejecución del comando `exec`

Luego, se cierra la terminal y con ella todos los procesos que se estaban ejecutando en ese momento. Se procede a abrir una nueva terminal y se cambia al usuario *labo3* mediante el uso del comando `su`. Ya una vez dentro de este usuario, se procede a dirigirse al directorio *home* de dicho usuario y se procede a realizar lo solicitado por el laboratorio.

Se utiliza el comando `chgrp` junto a los argumentos `grupolabo3` y `PruebasPermisos/`. La función de este comando es cambiar recursivamente de pertenencia del directorio creado `PruebasPermisos/` al grupo `grupolabo3`. Dicho procedimiento se puede ver documentado en la Figura 16.

```
fer9828@ushuaia:~$ pwd
/home/fer9828
fer9828@ushuaia:~$ su labo3
Password:
labo3@ushuaia:/home/fer9828$ cd ../
labo3@ushuaia:/home$ ls
fer9828 labo3 pcinfo
labo3@ushuaia:/home$ cd labo3/
labo3@ushuaia:~$ pwd
/home/labo3
labo3@ushuaia:~$ ls -l
total 4
drwxrwx--- 2 labo3 root 4096 Sep 23 17:02 PruebasPermisos
labo3@ushuaia:~$ sudo chgrp grupolabo3 PruebasPermisos/
[sudo] password for labo3:
labo3@ushuaia:~$ ls -l
total 4
drwxrwx--- 2 labo3 grupolabo3 4096 Sep 23 17:02 PruebasPermisos
labo3@ushuaia:~$
```

Figura 16: Cambio recursivo de la pertenencia del directorio `PruebasPermisos/`

Ahora, para probar que dicho cambio funcionó, se procede a realizar algunas pruebas. Como se observó anteriormente, el usuario `fer9828` no tenía acceso a ninguno de los permisos de lectura, ejecución y escritura dentro del directorio `PruebasPermisos/` debido a que se configuró que solo los usuarios del grupo `labo3` tuvieran acceso a dichos permisos.

Sin embargo, luego de agregar al usuario `fer9828` y el usuario `labo3` en el grupo `grupolabo3` y de cambiar recursivamente de pertenencia del directorio `PruebasPermisos/` al grupo `grupolabo3` ya el usuario `fer9828` obtendrá los permisos de escritura, lectura y ejecución del Directorio.

En la Figura 17. se puede apreciar como se ingresa desde el usuario `fer9828` al directorio `PruebasPermisos/` y efectivamente si se puede acceder al directorio sin problemas.

```
labo3@ushuaia:~$ su fer9828
Password:
fer9828@ushuaia:/home/labo3$ ls
PruebasPermisos
fer9828@ushuaia:/home/labo3$ cd PruebasPermisos/
fer9828@ushuaia:/home/labo3/PruebasPermisos$ ls
LEER.txt
fer9828@ushuaia:/home/labo3/PruebasPermisos$ nano LEER.txt
```

Figura 17: Prueba de permisos en directorio `PruebasPermisos`

En la Figura 18., mediante el uso del editor de texto nano se ingresa al archivo LEER.txt y se puede apreciar que ahora si es posible leer el texto dentro de LEER.txt.

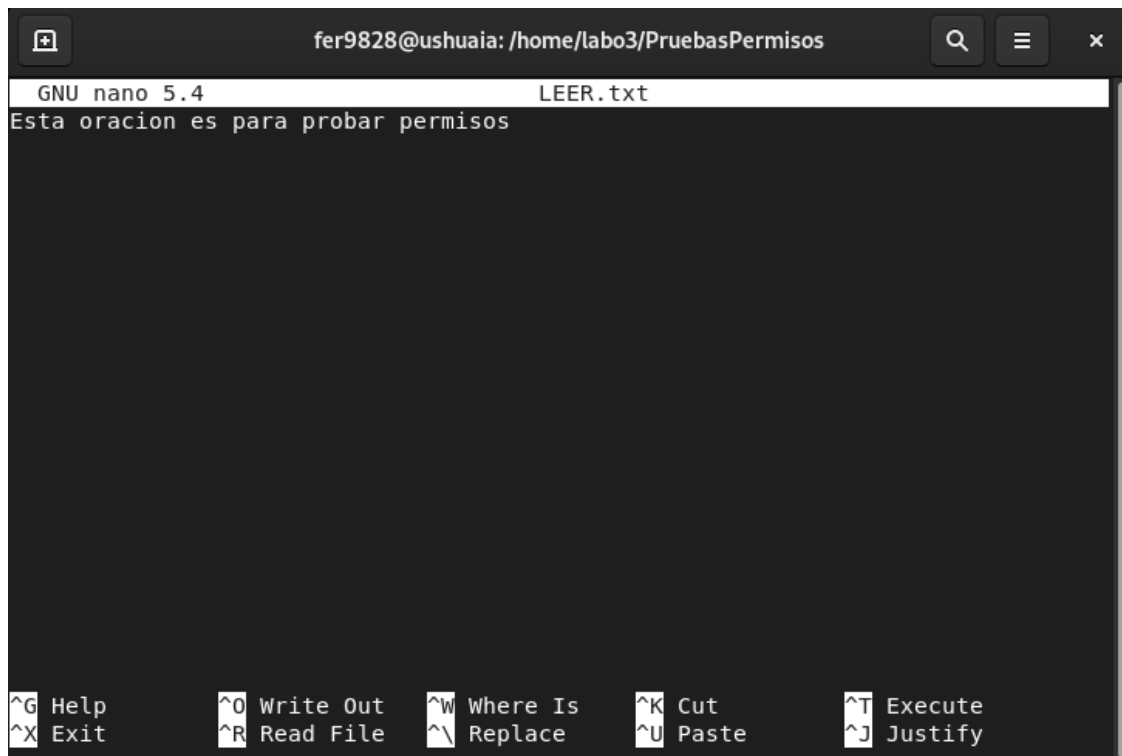


Figura 18: Lectura del texto de LEER.txt

Luego se debe eliminar el usuario *fer9828* del grupo *grupolabo3*. Para realizar este paso, se debe utilizar el comando *deluser* con los argumentos del nombre del usuario y el nombre del grupo del cual se desea eliminar. Este procedimiento se realizó desde el usuario *root* y se puede apreciar en la Figura 19. Además se puede comprobar mediante el comando *groups* que el usuario *fer9828* ya no pertenece al grupo *grupolabo3*

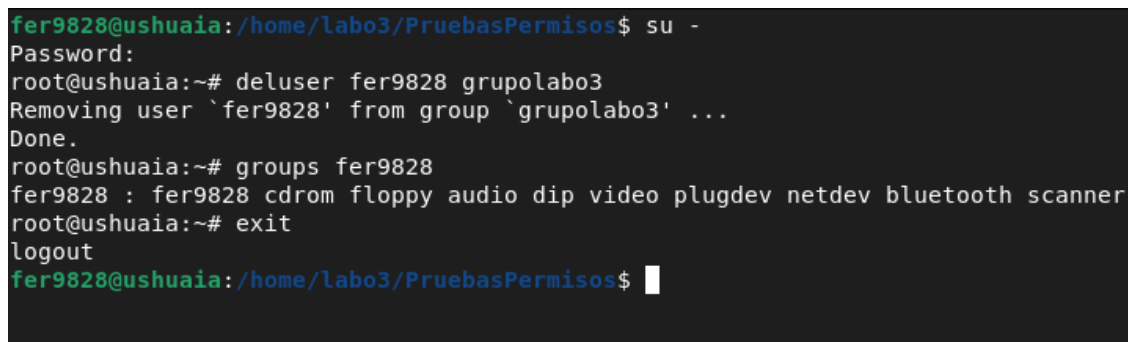


Figura 19: Eliminación del usuario *fer9828* del grupo *grupolabo3*

Finalmente, para terminar esta sección del laboratorio se ejecuta el comando *apt-get moo*

que imprime una vaca como se puede apreciar en la Figura 20.

```
fer9828@ushuaia:~$ apt-get moo
      (__)
     (oo)
  /-----\
 /  |      | \
*  \|----/\
   ~~~~
... "Have you mooed today?" ...
fer9828@ushuaia:~$
```

Figura 20: Ejecución del comando *apt-get moo*

3.2. Cron, crontab, rsync, ssh

Es importante notar antes de desarrollar esta parte del laboratorio que la investigación teórica sobre la definición de *cron* y el formato de *crontab* y sus usos y el uso del comando *rsync* ya fue realizada previamente y se encuentra en la Nota Teórica de este laboratorio.

Se procede a realizar el paso de crear llaves para el usuario *fer9828* de modo que no se pida la contraseña a la hora de conectarse con el servidor *tarcoles.eie.ucr.ac.cr*. Primero, se procede a escribir el comando *ssh-keygen -t rsa*. El comando *ssh-keygen* se utiliza para la creación, gestión y conversión de claves utilizadas para la autenticación de cliente y servidor. Mientras que los argumentos, *-t rsa* se utiliza para especificar que se está creando una llave de tipo RSA.

```
fer9828@ushuaia:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/fer9828/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/fer9828/.ssh/id_rsa
Your public key has been saved in /home/fer9828/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:PVBfQjJsbhE/kcLYilyQRPFjMBov1nLL1vi+5FcZexU fer9828@ushuaia
The key's randomart image is:
+----[RSA 3072]-----+
|  .+%.* ==0+.. |
|  +.+=.B=.+ E |
|  . *=B 0+ . |
|  0.*++ .. |
|  . .Soo +. |
|  .. +. |
|  .. . |
|  0. . |
|  oo |
+-----[SHA256]-----+
fer9828@ushuaia:~$
```

Figura 21: Creación de la llave RSA

Ya una vez creada la llave, se procede a configurarla en el servidor *tarcoles.eie.ucr.ac.cr*. Para esto, se utiliza el comando *ssh-copy-id* en conjunto al usuario ya creado en dicho servidor. Se puede observar en la Figura 22. y en la Figura 23. que el proceso fue realizado con éxito.

```
fer9828@ushuaia:~$ ssh-copy-id fjjimenez@tarcoles.eie.ucr.ac.cr
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
fjjimenez@tarcoles.eie.ucr.ac.cr's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'fjjimenez@tarcoles.eie.ucr.ac.cr'"
and check to make sure that only the key(s) you wanted were added.

fer9828@ushuaia:~$
```

Figura 22: Configuración de la llave con el servidor *tarcoles.eie.ucr.ac.cr*.

```
fer9828@ushuaia:~$ ssh fjjimenez@tarcoles.eie.ucr.ac.cr
Linux tarcoles 4.19.0-17-amd64 #1 SMP Debian 4.19.194-2 (2021-06-21) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Sep 12 09:56:36 2021 from 201.202.14.22
fjjimenez@tarcoles:~$ exit
logout
Connection to tarcoles.eie.ucr.ac.cr closed.
fer9828@ushuaia:~$
```

Figura 23: Llave exitosa en el servidor *tarcoles.eie.ucr.ac.cr*.

Se procede a copiar el directorio *PruebasPermisos/* en el \$HOME del usuario *fer9828*. Este procedimiento se realiza utilizando el comando *cp -r* y se puede apreciar en la Figura 24.

```
fer9828@ushuaia:~/home/labo3$ ls
PruebasPermisos
fer9828@ushuaia:~/home/labo3$ cp -r PruebasPermisos/ ../fer9828/
fer9828@ushuaia:~/home/labo3$ cd ../fer9828/ ls
bash: cd: too many arguments
fer9828@ushuaia:~/home/labo3$ cd ../fer9828/
fer9828@ushuaia:~$ ls
Desktop  Downloads  Pictures  primera.txt  prueba.sh      Public  variables_especiales.sh
Documents Music      Primer    primer.c     PruebasPermisos Templates Videos
fer9828@ushuaia:~$ pwd
/home/fer9828
```

Figura 24: Copia del directorio *PruebasPermisos/*

Seguidamente, se desea respaldar el directorio *PruebasPermisos/* en el servidor. Dicho respaldo se desea realizar en el directorio \$HOME del usuario en el servidor. Para realizar este respaldo utilizando rsync mediante ssh, se utiliza el comando *rsync avz e ssh /home/fer9828/PruebasPermisos/ fjjimenez@tarcoles.eie.ucr.ac.cr:/home/fjjimenez/PruebasPermisos/*

Esto quiere decir que se hará un respaldo desde una carpeta local hacia un host remoto en el directorio \$HOME del usuario en el servidor.

```

fer9828@ushuaia:~/PruebasPermisos$ ssh fjjimenez@tarcoles.eie.ucr.ac.cr
Linux tarcoles 4.19.0-17-amd64 #1 SMP Debian 4.19.194-2 (2021-06-21) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Sep 25 07:00:55 2021 from 201.202.14.125
fjjimenez@tarcoles:~$ pwd
/home/fjjimenez
fjjimenez@tarcoles:~$ exit
logout
Connection to tarcoles.eie.ucr.ac.cr closed.
fer9828@ushuaia:~/PruebasPermisos$ rsync -avz -e ssh /home/fer9828/PruebasPermisos/ fjjimenez@tarcoles.eie.ucr.ac.cr:/home/fjjimenez/PruebasPermisos/
sending incremental file list
created directory /home/fjjimenez/PruebasPermisos
./
LEER.txt

sent 165 bytes  received 93 bytes  516.00 bytes/sec
total size is 37  speedup is 0.14
fer9828@ushuaia:~/PruebasPermisos$

```

Figura 25: Creación del respaldo PruebasPermisos/ en el servidor

Se puede comprobar ingresando al servidor que efectivamente si se logró crear un respaldo del directorio dentro del mismo. Dicho proceso se puede apreciar en la Figura 26.

```

fer9828@ushuaia:~/PruebasPermisos$ ssh fjjimenez@tarcoles.eie.ucr.ac.cr
Linux tarcoles 4.19.0-17-amd64 #1 SMP Debian 4.19.194-2 (2021-06-21) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Sep 25 10:08:50 2021 from 201.202.14.125
fjjimenez@tarcoles:~$ pwd
/home/fjjimenez
fjjimenez@tarcoles:~$ ls
PruebasPermisos
fjjimenez@tarcoles:~$ cd PruebasPermisos/
fjjimenez@tarcoles:~/PruebasPermisos$ ls -l
total 4
-rw-r--r-- 1 fjjimenez domusers 37 Sep 24 22:27 LEER.txt
fjjimenez@tarcoles:~/PruebasPermisos$

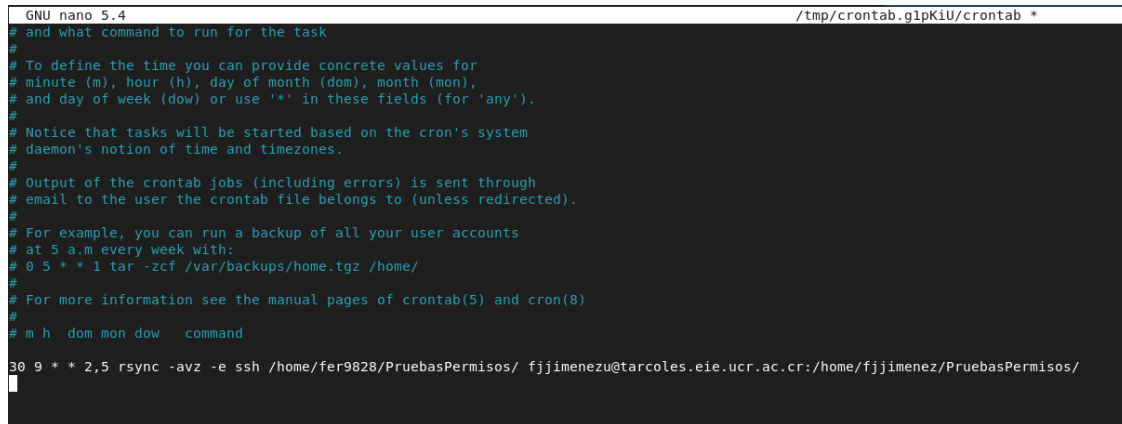
```

Figura 26: Creación del respaldo PruebasPermisos/ en el servidor

Finalmente para terminar esta sección, se debe programar la computadora para realizar este proceso de respaldo todos los Martes y Viernes a las 9:30 a.m.. Para lograr esto, se utiliza el comando *crontab -e*. Dicho comando abrirá un archivo de texto editable donde se podrá agregar la tarea que se desea automatizar, en este caso: El respaldo del directorio *PruebasPermisos/* en el servidor.

Una vez dentro del editor de texto como se puede apreciar en la Figura 27., se procede a escribir la instrucción para que se ejecute automáticamente en los períodos seleccionados.

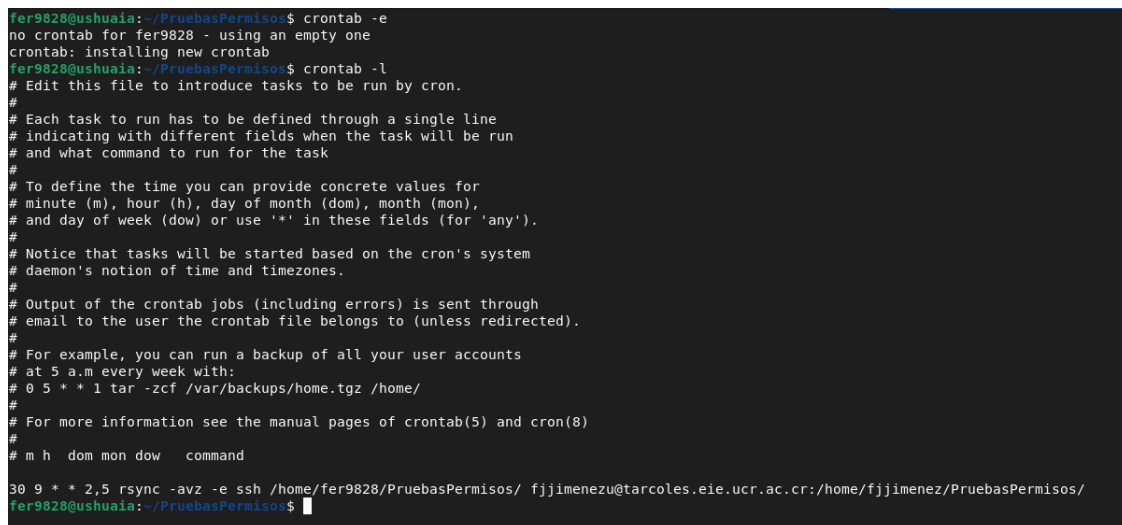
El formato para indicar la instrucción es la siguiente: “minutos” “horas” “día del mes” “mes” “día de la semana” “usuario” “comando o script”. Por lo que para lograr lo deseado en este laboratorio se configuró: “30” “9” “*” “*” “2,5”. Por lo que se desea que todos los Martes y Viernes de todos los meses a las 9:30 horas se ejecute el comando de respaldo automáticamente. Cabe destacar que los asteriscos significan que el proceso se hará cada” minuto, hora, día de mes, mes o día de la semana depende de la posición del mismo.



```
GNU nano 5.4 /tmp/crontab.glpKiU/crontab *
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow command
30 9 * * 2,5 rsync -avz -e ssh /home/fer9828/PruebasPermisos/ fjjimenezu@tarcoles.eie.ucr.ac.cr:/home/fjjimenez/PruebasPermisos/
```

Figura 27: Creación de la instrucción *cron*

Se puede comprobar que se logró el proceso de manera correcta si se utiliza el comando *crontab -l* cuya función es mostrar en pantalla los procesos automatizados activos en el sistema. Se puede concluir que si se guardó correctamente el proceso.



```
fer9828@ushuaia:~/PruebasPermisos$ crontab -e
no crontab for fer9828 - using an empty one
crontab: installing new crontab
fer9828@ushuaia:~/PruebasPermisos$ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow command
30 9 * * 2,5 rsync -avz -e ssh /home/fer9828/PruebasPermisos/ fjjimenezu@tarcoles.eie.ucr.ac.cr:/home/fjjimenez/PruebasPermisos/
fer9828@ushuaia:~/PruebasPermisos$
```

Figura 28: Automatización mediante *crontab* realizada con éxito

3.3. Instalación de programas desde código fuente

Ahora, se procede a realizar la última parte del laboratorio que consiste en instalar un programa desde su código fuente. Dicho programa es *NodeJS* y su instalación no incluirá paquetes del repositorio de la distribución, todo será desde su código fuente.

Primero se debe ingresar al enlace adjuntado en el laboratorio: <https://github.com/nodejs/node> y se procede a copiar la URL para descargar los archivos a través del terminal con el comando `wget`

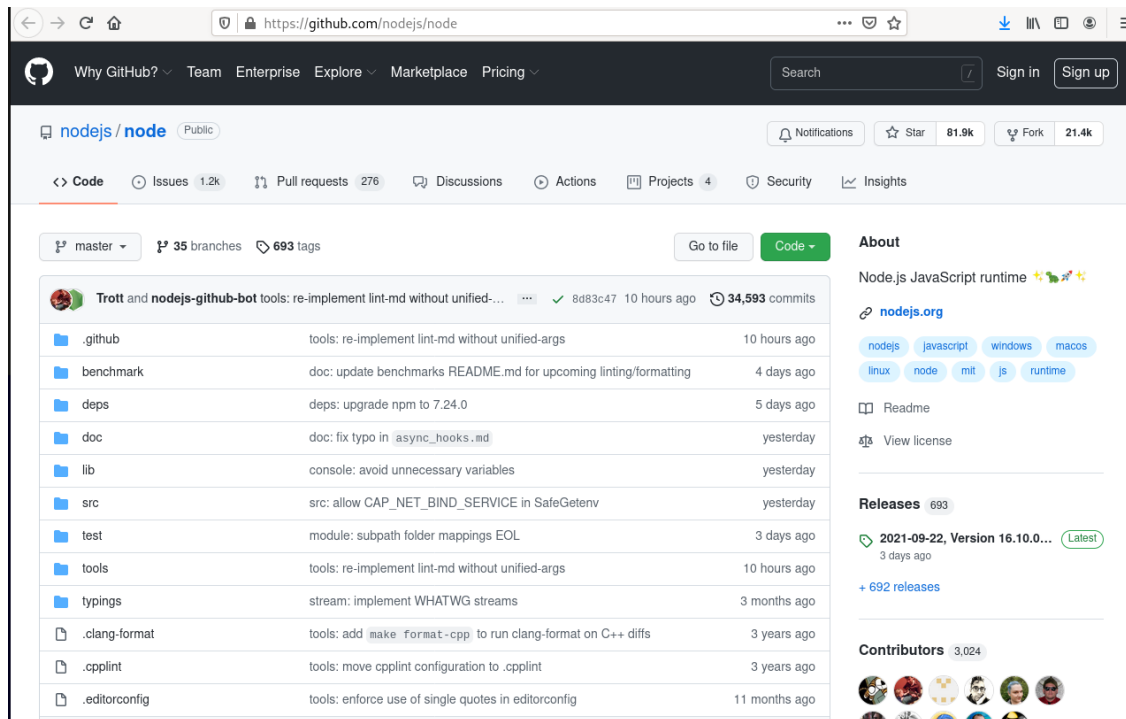


Figura 29: Ingreso y copia del URL desde GitHub

Ya una vez copiado el URL se procede a utilizar el comando `wget` en la terminal de la consola y seguido de eso, se pega el URL que se había copiado recientemente. Este comando se utiliza para descargar los archivos de códigos fuente del programa.

```
fer9828@ushuaia:~/Downloads$ wget https://github.com/nodejs/node.git
--2021-09-25 08:11:00-- https://github.com/nodejs/node.git
Resolving github.com (github.com)... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://github.com/nodejs/node [following]
--2021-09-25 08:11:01-- https://github.com/nodejs/node
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'node.git'

node.git          [  <=>          ] 323.02K  785KB/s   in 0.4s

2021-09-25 08:11:02 (785 KB/s) - 'node.git' saved [330776]

fer9828@ushuaia:~/Downloads$ ls
node.git
```

Figura 30: Uso del comando *wget* en la terminal

Luego, se debe ejecutar el comando *git clone* para poder descargar el código fuente del programa hacia el sistema Linux que se está utilizando. Es importante notar que para poder utilizar el comando *git*, se llevo a cabo una instalación previa de los paquetes en la terminal mediante el comando *sudo apt install git*. Ya una vez ejecutado el comando, se puede apreciar en la Figura 30. que mediante el comando *ls-l* se observa el archivo fue descomprimido con éxito.

```
fer9828@ushuaia:~/Downloads$ git clone https://github.com/nodejs/node.git
Cloning into 'node'...
remote: Enumerating objects: 698247, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 698247 (delta 0), reused 0 (delta 0), pack-reused 698246
Receiving objects: 100% (698247/698247), 712.40 MiB | 3.06 MiB/s, done.
Resolving deltas: 100% (521445/521445), done.
Updating files: 100% (32707/32707), done.
fer9828@ushuaia:~/Downloads$ ls
node node.git
fer9828@ushuaia:~/Downloads$
```

Figura 31: Archivo .zip descomprimido correctamente

Seguidamente, una vez dentro del directorio creado *node*, se utiliza el script *./configure*, cuya principal función es localizar el código fuente del paquete descargado y así que pueda ser cargado y compilado en Linux.

```

fer9828@ushuaia:~/Downloads$ cd node/
fer9828@ushuaia:~/Downloads/node$ ls
android-configure  common.gypi      lib              src
AUTHORS            configure         LICENSE          test
benchmark          configure.py     Makefile         tools
BSDmakefile        CONTRIBUTING.md  node.gyp         tsconfig.json
BUILDING.md         deps             node.gypi        typings
CHANGELOG.md        doc              onboarding.md   vcbuild.bat
codecov.yml         glossary.md      README.md
CODE_OF_CONDUCT.md GOVERNANCE.md    SECURITY.md
fer9828@ushuaia:~/Downloads/node$ sudo ./configure
[sudo] password for fer9828:
Node.js configure: Found Python 3.9.2...
INFO: configure completed successfully
fer9828@ushuaia:~/Downloads/node$

```

Figura 32: Uso del script `./configure`

Como se pudo observar, se localizó el código fuente con éxito por lo que se procede primero a utilizar el comando *make clean* cuya función principal es limpiar archivos innecesarios antes de poder utilizar el comando *make*.

```

fer9828@ushuaia:~/Downloads/node$ sudo make clean
rm -f -r out/Makefile node node_g out/Release/node \
    out/Release/node.exp
rm -f -r node_modules
rm -f test.tap
make testclean
rm -f -r test/tmp*
rm -f -r test/.tmp*
make test-addons-clean
rm -f -r test/addons/??_*/
rm -f -r test/addons/*/build
rm -f test/addons/.buildstamp test/addons/.docbuildstamp
make test-js-native-api-clean
rm -f -r test/js-native-api/*/build
rm -f test/js-native-api/.buildstamp
make test-node-api-clean
rm -f -r test/node-api/*/build
rm -f test/node-api/.buildstamp
make bench-addons-clean
rm -f -r benchmark/napi/*/build
rm -f benchmark/napi/.buildstamp
fer9828@ushuaia:~/Downloads/node$

```

Figura 33: Uso del comando *make clean*

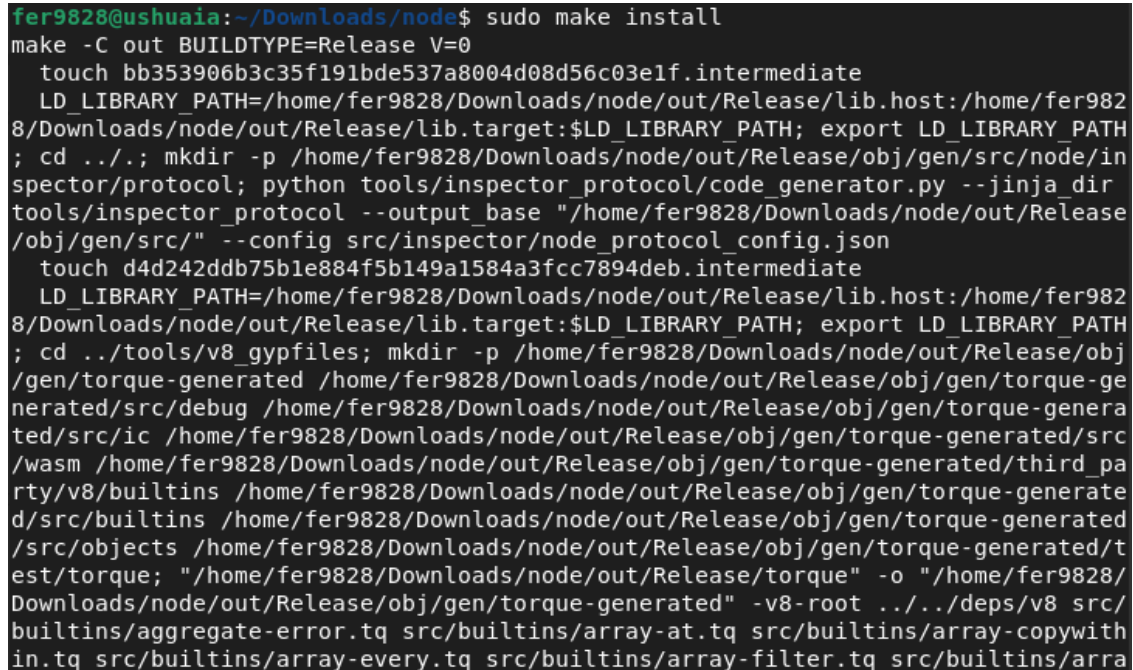
Ya una vez configurado esto, se procede a utilizar el comando *make*. El comando *make* se encarga de ayudar a compilar e instalar muchas utilidades de código abierto a través de la línea de comandos.



```
fer9828@ushuaia:~/Downloads/node$ sudo make
```

Figura 34: Uso del comando *make*

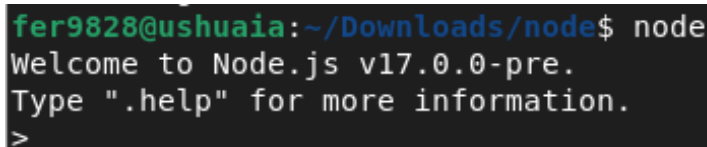
Luego de un largo tiempo, se finaliza con la configuración de diferentes tipos de archivos. Luego como se aprecia en la Figura 35., se procede a utilizar el comando *make install* cuya función es copiar los archivos compilados por el comando *make* a diferentes rutas del sistema.



```
fer9828@ushuaia:~/Downloads/node$ sudo make install
make -C out BUILDTYPE=Release V=0
touch bb353906b3c35f191bde537a8004d08d56c03e1f.intermediate
LD_LIBRARY_PATH=/home/fer9828/Downloads/node/out/Release/lib.host:/home/fer9828/Downloads/node/out/Release/lib.target:$LD_LIBRARY_PATH; export LD_LIBRARY_PATH; cd ../../; mkdir -p /home/fer9828/Downloads/node/out/Release/obj/gen/src/node/inspector/protocol; python tools/inspector_protocol/code_generator.py --jinja_dir tools/inspector_protocol --output_base "/home/fer9828/Downloads/node/out/Release/obj/gen/src/" --config src/inspector/node_protocol_config.json
touch d4d242ddb75b1e884f5b149a1584a3fcc7894deb.intermediate
LD_LIBRARY_PATH=/home/fer9828/Downloads/node/out/Release/lib.host:/home/fer9828/Downloads/node/out/Release/lib.target:$LD_LIBRARY_PATH; export LD_LIBRARY_PATH; cd ../tools/v8_gypfiles; mkdir -p /home/fer9828/Downloads/node/out/Release/obj/gen/torque-generated /home/fer9828/Downloads/node/out/Release/obj/gen/torque-generated/src/debug /home/fer9828/Downloads/node/out/Release/obj/gen/torque-generated/src/ic /home/fer9828/Downloads/node/out/Release/obj/gen/torque-generated/src/wasm /home/fer9828/Downloads/node/out/Release/obj/gen/torque-generated/third_party/v8/builtins /home/fer9828/Downloads/node/out/Release/obj/gen/torque-generated/src/builtins /home/fer9828/Downloads/node/out/Release/obj/gen/torque-generated/src/objects /home/fer9828/Downloads/node/out/Release/obj/gen/torque-generated/torque; "/home/fer9828/Downloads/node/out/Release/torque" -o "/home/fer9828/Downloads/node/out/Release/obj/gen/torque-generated" -v8-root ../../deps/v8 src/builtins/aggregate-error.tq src/builtins/array-at.tq src/builtins/array-copywith.in.tq src/builtins/array-every.tq src/builtins/array-filter.tq src/builtins/arra
```

Figura 35: Uso del comando *make install*

Una vez terminado esto, se puede observar que si se ingresa la palabra *node*, se dará la bienvenida al programa lo que significa que fue instalado con éxito.



```
fer9828@ushuaia:~/Downloads/node$ node
Welcome to Node.js v17.0.0-pre.
Type ".help" for more information.
>
```

Figura 36: Ejecución del programa

Ya una vez probado el programa, se procede a desinstalar el mismo. Esto se realiza mediante el comando *sudo make uninstall*

```
fer9828@ushuaia:~/Downloads/node$ sudo make uninstall
/usr/bin/python3.9 tools/install.py uninstall '' '/usr/local'
removing /usr/local/bin/node
removing /usr/local/share/systemtap/tapset/node.stp
removing /usr/local/share/doc/node/gdbinit
removing /usr/local/share/doc/node/lldb_commands.py
removing /usr/local/share/man/man1/node.1
removing /usr/local/lib/node_modules/npm/README.md
removing /usr/local/lib/node_modules/npm/package.json
removing /usr/local/lib/node_modules/npm/.npmrc
removing /usr/local/lib/node_modules/npm/LICENSE
removing /usr/local/lib/node_modules/npm/lib/npm.js
removing /usr/local/lib/node_modules/npm/lib/stars.js
removing /usr/local/lib/node_modules/npm/lib/token.js
removing /usr/local/lib/node_modules/npm/lib/root.js
removing /usr/local/lib/node_modules/npm/lib/access.js
removing /usr/local/lib/node_modules/npm/lib/set-script.js
removing /usr/local/lib/node_modules/npm/lib/docs.js
removing /usr/local/lib/node_modules/npm/lib/exec.js
removing /usr/local/lib/node_modules/npm/lib/restart.js
removing /usr/local/lib/node_modules/npm/lib/bugs.js
removing /usr/local/lib/node_modules/npm/lib/view.js
removing /usr/local/lib/node_modules/npm/lib/ci.js
```

Figura 37: Ejecución del programa

Finalmente, para comprobar que el programa ya fue desinstalado, se procede a escribir de nuevo la palabra *node* y se puede apreciar que el sistema nos señala que ya no existe.

```
fer9828@ushuaia:~/Downloads/node$ node
bash: /usr/local/bin/node: No such file or directory
fer9828@ushuaia:~/Downloads/node$
```

Figura 38: Programa desinstalado correctamente

4. Conclusiones y recomendaciones

El desarrollo de este laboratorio fue de gran importancia para seguir desarrollando conocimiento de la manipulación a fondo de un sistema operativo. Esta vez se logró explorar en qué consiste la creación de un usuario, los grupos y la administración de sus respectivos de permisos de lectura, escritura y ejecución.

En la segunda parte se logró adquirir más conocimientos sobre el funcionamiento y utilidad de los servidores. Además se logró crear una llave RSA para facilitar el acceso al servidor y así poder mejorar el flujo de trabajo ya que no estar escribiendo una contraseña cada vez que se ingresa al servidor ahorra bastante tiempo a largo plazo.

Finalmente, se logró instalar un programa desde su código fuente. Dicho procedimiento fue de gran importancia debido a que en un sistema operativo como Linux que es de software libre, poder tener acceso al código fuente es algo vital debido a que eso permite realizar cambios necesarios o mejoras a los programas haciendo uso de su código fuente.

5. Referencias

- [1] Oriol, L. (2018). El Usuario Root y el Control de Privilegios en Linux: Sudo, Su y PolicyKit. ComputerNewAge. Retrieved 20 September 2021, from <https://computernewage.com/2018/04/07/linux-root-control-privilegios-sudo-su-policykit/control-privilegios>.
- [2] Oriol, L. (2015). Conoce la Arquitectura de Permisos de Linux al Detalle. ComputerNewAge. Retrieved 20 September 2021, from <https://computernewage.com/2015/06/27/conoce-la-estructura-de-permisos-de-linux-al-detalle/>.
- [3] Garita, A. (2013). Cron crontab, explicados. Desde Linux. Retrieved 20 September 2021, from https://blog.desdelinux.net/cron-crontab-explicados/Que_es_cron.
- [4] G.B.(2020, Setiembre, 23). Comando Rsync de Linux(sincronización remota)”. Recuperado de <https://www.hostinger.es/tutoriales/rsync-linux>
- [5] J.Ellingwood.(2020,Diciembre,15).Como usar Rsync para sincronizar directorios locales y remotos”. Recuperado de: <https://www.digitalocean.com/community/tutorials/how-to-use-rsync-to-sync-local-and-remote-directories-es>
- [6] Rojas, I. (2019). Linux: navegando por los mares del código fuente - ArchiTecnología. ArchiTecnología. Retrieved 20 September 2021, from <https://architecnologia.es/linux-navegando-por-los-mares-del-codigo-fuente>.
- [7] Menjívar, M. (2021). Explicación de los grupos de usuarios en Linux: cómo agregar un nuevo grupo, agregar un nuevo miembro, y cambiar de grupo. freeCodeCamp.org. Retrieved 23 September 2021, from <https://www.freecodecamp.org/espanol/news/explicacion-de-los-grupos-de-usuarios-en-linux/>.
- [8] Rodriguez, C. (2019). La función exec. Campusvirtual.ull.es. Retrieved 23 September 2021, from https://campusvirtual.ull.es/ocw/pluginfile.php/2170/mod_resource/content/0/perlexamples/